Karlsruher Institut für Technologie

Algorithmen für Routenplanung

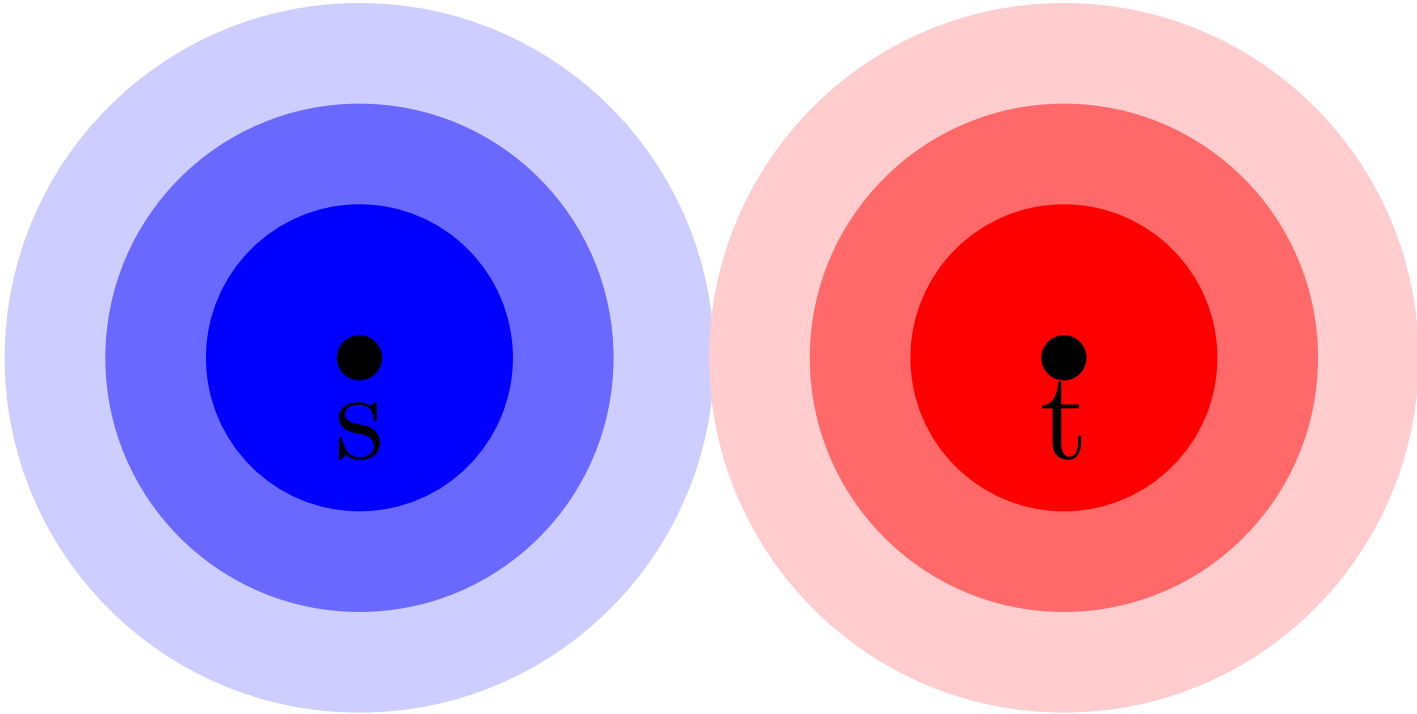
6. Vorlesung, Sommersemester 2010

Daniel Delling | 17. Mai 2010

MICROSOFT RESEARCH SILICON VALLEY

Letztes Mal

- RE-based Routing
- Kombination mit ALT
- Contraction Hierarchies



Zusammenfassung letztes Mal

- RE-based Routing
- Kombination mit ALT
- Contraction Hierarchies

Gemeinsamkeiten RE/CH:

- hierarchisch
- suchen nur aufwärts

Themen Heute

- Many-to-Many Route Planning
  - berechnet Distanz-Matrix
- Alternativ-Routen in Straßennetzen
  - Definition
  - schnelle Berechnung

Many-to-Many Kürzeste Wege

Gegeben:

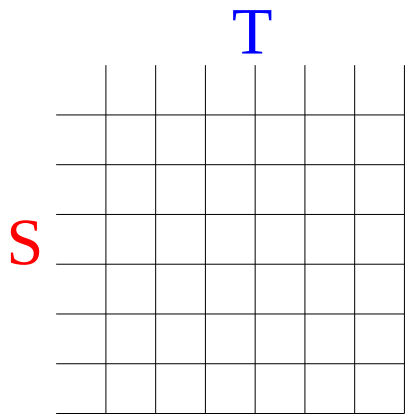
- Graph
- Knotenmengen $S, T \in V$

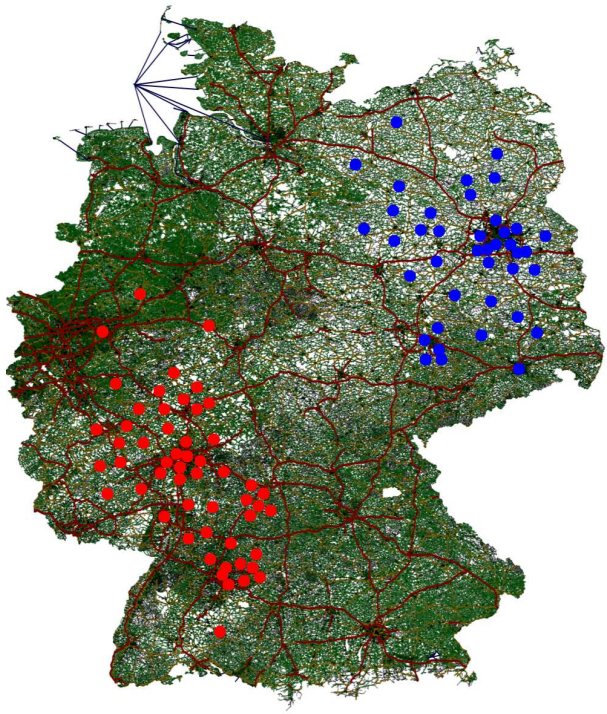
Gesucht:

- Distanzmatrix D

Anwendungen:

- vehicle routing
- traveling salesman





Naiver Ansatz

Erster Ansatz:

- führe $|S|$ mal Dijkstra aus
- führe $|S| \times |T|$ mal die Query einer Beschleunigungstechnik aus

Problem:

- typische Eingabegröße (Europa, 10 000 x 10 000 Einträge)
- Variante 1 dauert ca. 1 Tag
- Variante 2 dauert ca. 2-3 Stunden

Ungerichtete Bidir. Techniken

Definition:

- $\overrightarrow{\sigma}(s,t)$: Suchraum der Vorwärtssuche von s nach t
- $\overleftarrow{\sigma}(s,t)$ analog
- eine bidirektionale Suche ist Ziel-unabhängig, gdw.

$$\forall (s,t_1,t_2) \in V^3 : \overrightarrow{\sigma}(s,t_1) = \overrightarrow{\sigma}(s,t_2) \quad \text{und}$$
$$\forall (s_1,s_2,t) \in V^3 : \overleftarrow{\sigma}(s_1,t) = \overleftarrow{\sigma}(s_2,t)$$

Ungerichtete Bidir. Techniken

Definition:

- $\overrightarrow{\sigma}(s,t)$: Suchraum der Vorwärtssuche von s nach t
- $\overleftarrow{\sigma}(s,t)$ analog
- eine bidirektionale Suche ist Ziel-unabhängig, gdw.

$$\forall (s,t_1,t_2) \in V^3 : \overrightarrow{\sigma}(s,t_1) = \overrightarrow{\sigma}(s,t_2) \quad \text{und}$$
$$\forall (s_1,s_2,t) \in V^3 : \overleftarrow{\sigma}(s_1,t) = \overleftarrow{\sigma}(s_2,t)$$

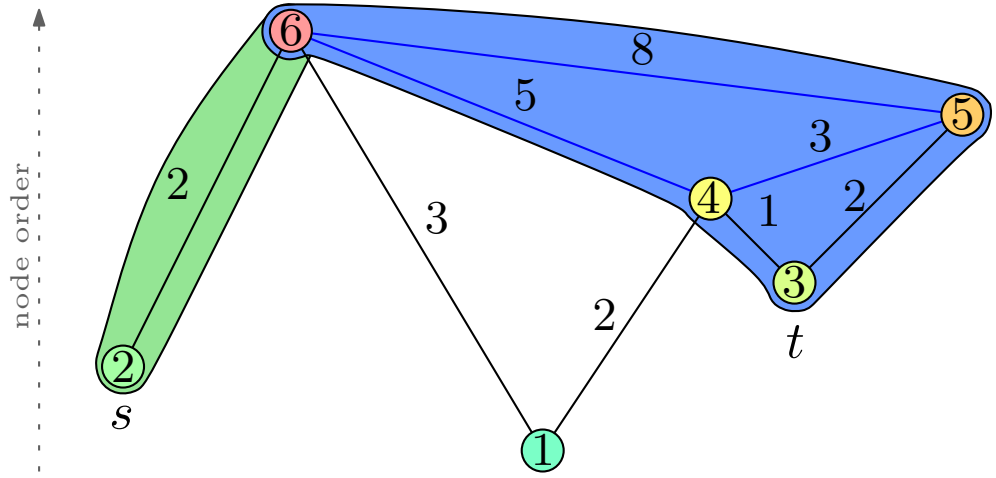
Beispiele:

- Bidirektionaler Dijkstra
- ohne Stoppkriterium, lass laufen bis Queues leer sind

Hierarchische Techniken

Beobachtung:

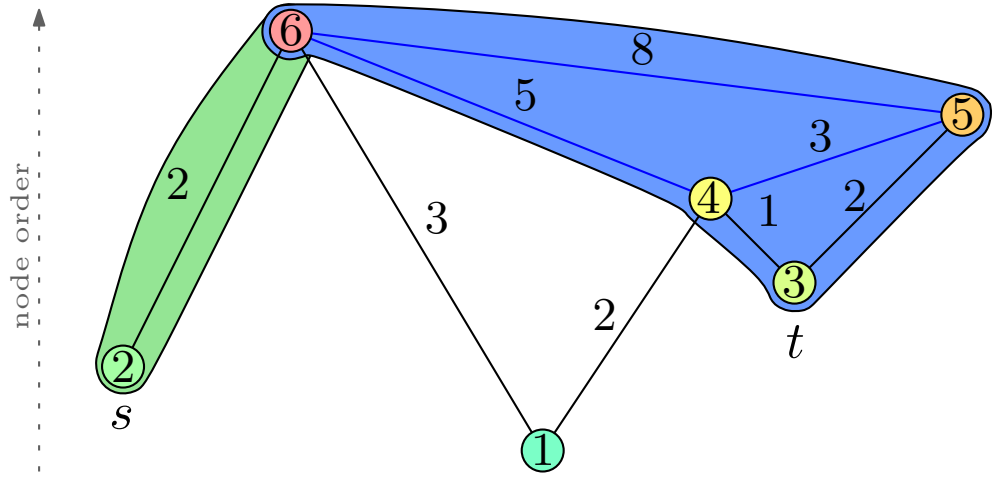
- suchen nur aufwärts
- sind nicht zielgerichtet



Hierarchische Techniken

Beobachtung:

- suchen nur aufwärts
- sind nicht zielgerichtet



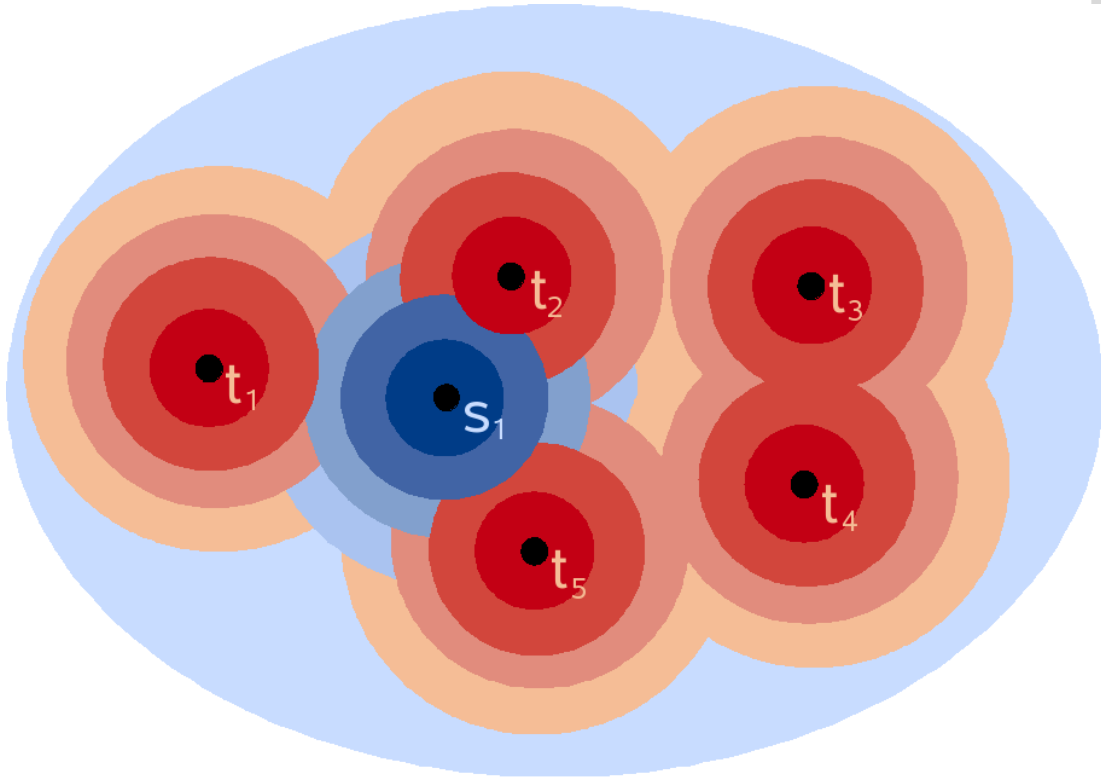
somit:

- Bidirektionaler Dijkstra
- Reach/RE
- Contraction Hierarchies
- ohne Stoppkriterium, lass laufen bis Queues leer sind

Vorgehen

Idee:

- führe $|T|$ Rückwärtssuchen aus
- speicher für jedes u die Abstände zu allen $t \in T$
- verwalte temporäre Distanztabelle D
- führe $|S|$ Vorwärtssuchen aus
- aktualisiere Einträge in D



Problem:

- verwalten der Suchräume?

Schneiden der Suchräume

während Rückwärtssuchen:

- breche nicht ab
- für jedes erreichte u:
  - füge Element $(u, t, d(u, t))$ in einen Vektor ein

Kompression:

- sortiere Einträge nach u
- speichere für jeden Knoten u einen Bucket $\beta(u)$ mit allen $(t, d(u, t))$ ab
- mittels Adjazenzarray

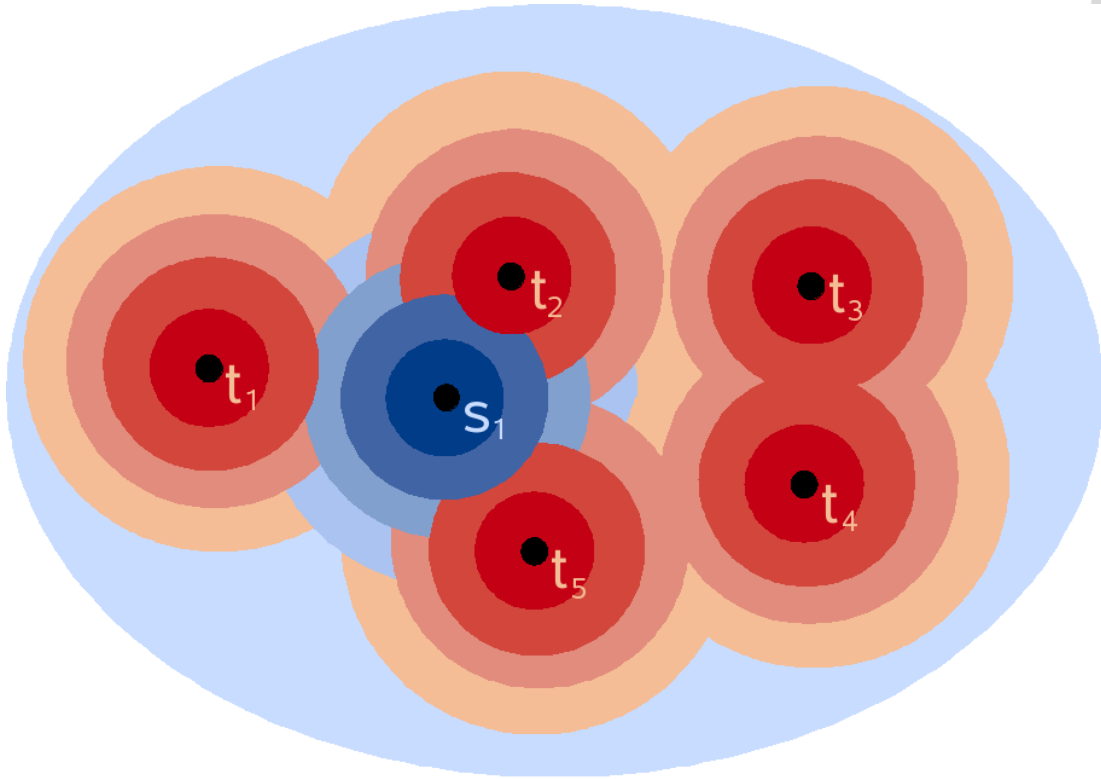
während Vorwärtssuchen:

- breche nicht ab
- für jedes erreichte u:
  - durchsuche Bucket $\beta(u)$
  - aktualisiere Distanzmatrix

Asymmetrie

Beobachtungen:

- Durchsuchen der Buckets dominiert die Laufzeit



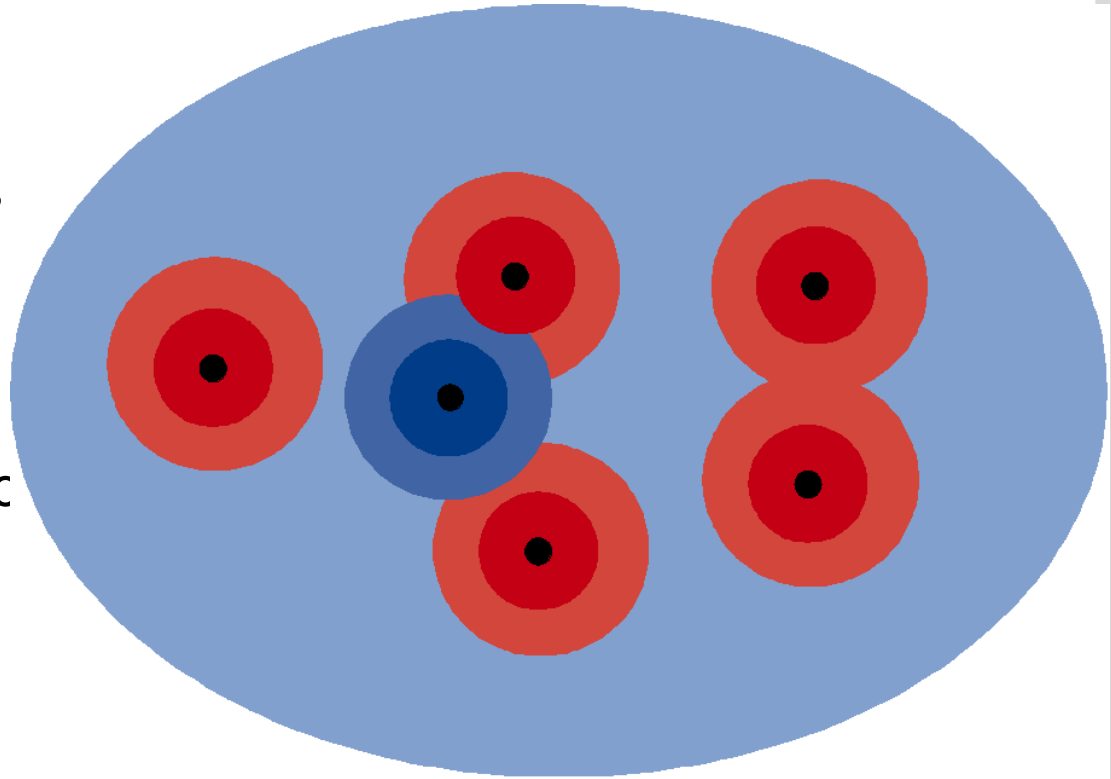
Asymmetrie

Beobachtungen:

- Durchsuchen der Buckets dominiert die Laufzeit

Idee:

- Rückwärtssuche durchsuc
  nicht obersten Level
- schneide Hierarchie oben
- dadurch weniger Buckets
- aber: Vorwärtssuche dauern länger



Pfadentpackung

Vorgehen:

- speichere jeden Suchbaum von $t \in T$
- speichere jeden Suchbaum von $s \in S$
- für jeden Eintrag in D halte Zeiger auf Knoten in den beiden Suchbäumen, die zu s und t gehören, vor

Optimierung:

- schneide Teile der Suchbäume weg, die man nicht benötigt

Ergebnisse M2M Europa

Setup:

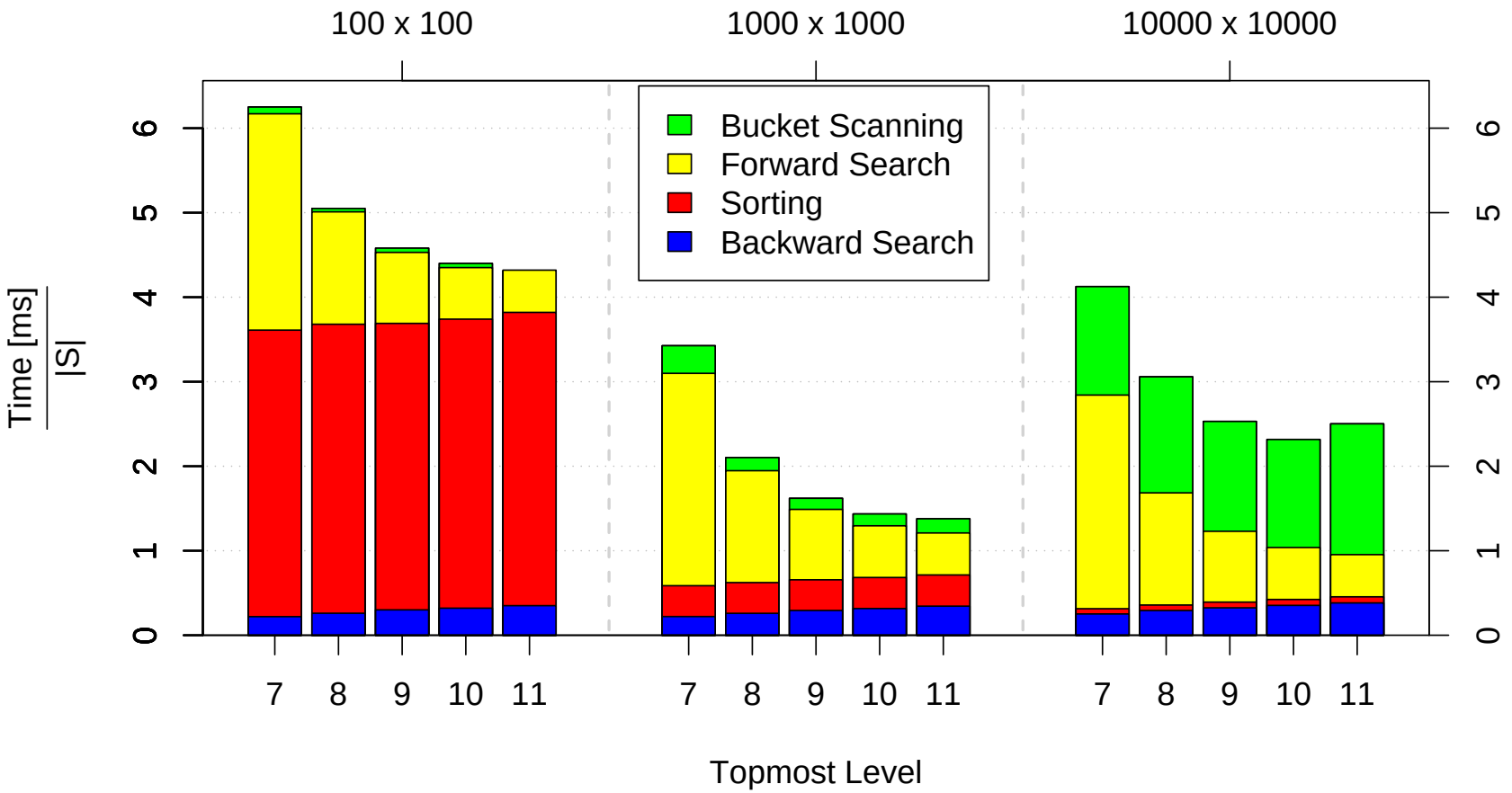
- Eingabe Europa
- berechne $S \times S$ Distanztabelle
- Laufzeit in Sekunden

$\lvert S \rvert$	100	500	1 000	5 000	10 000	20 000
CH	0.4	0.5	0.6	3.3	10.2	36.6

Beobachtung:

- 10 000 Matrix in 10,2 Sekunden berechenbar
- pro Eintrag: 0.1 μs

Ergebnisse M2M Details



Wiederholung Punkt-zu-Punkt

Anfrage:

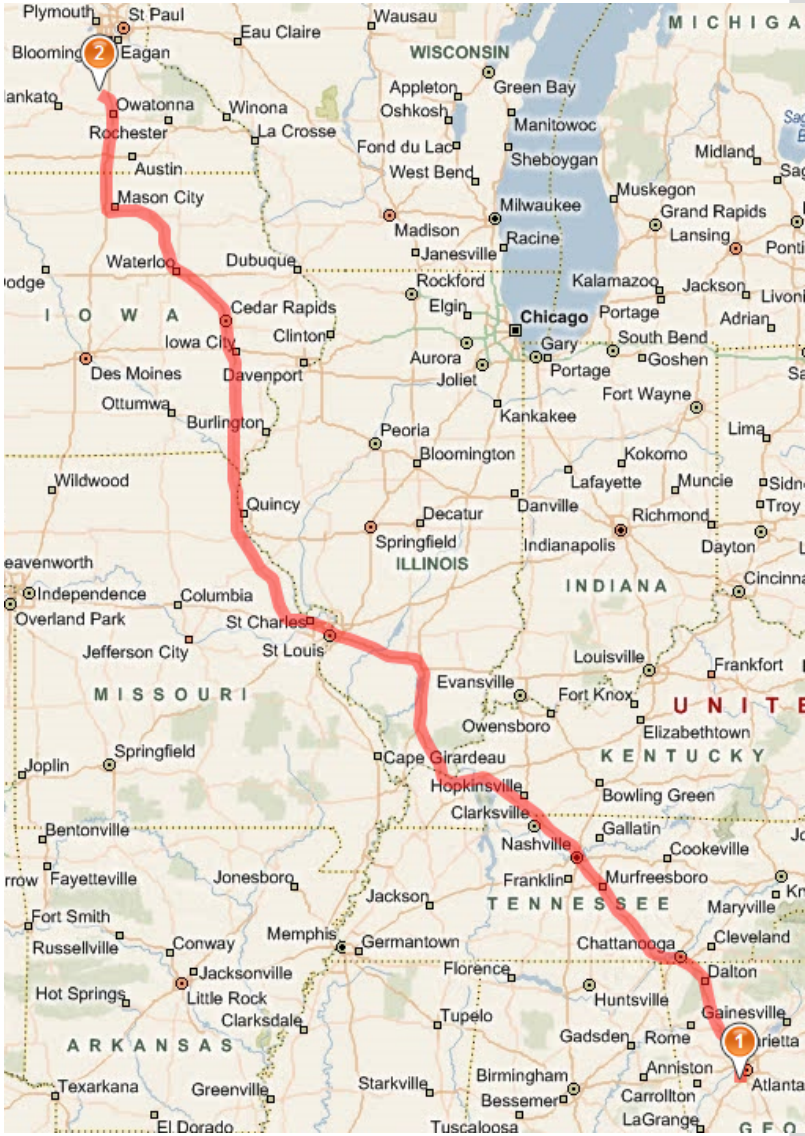
- finde die beste Route in einem Transportnetz

Idee:

- Netzwerk als Graph $G = (V, E)$
- Kantengewichte sind Reisezeiten
- kürzester Weg in G entspricht schnellster Verbindung

Ergebnisse:

- schnelle Algorithmen existieren (diese Vorlesung)



Alternativ-Routen

Anfrage:

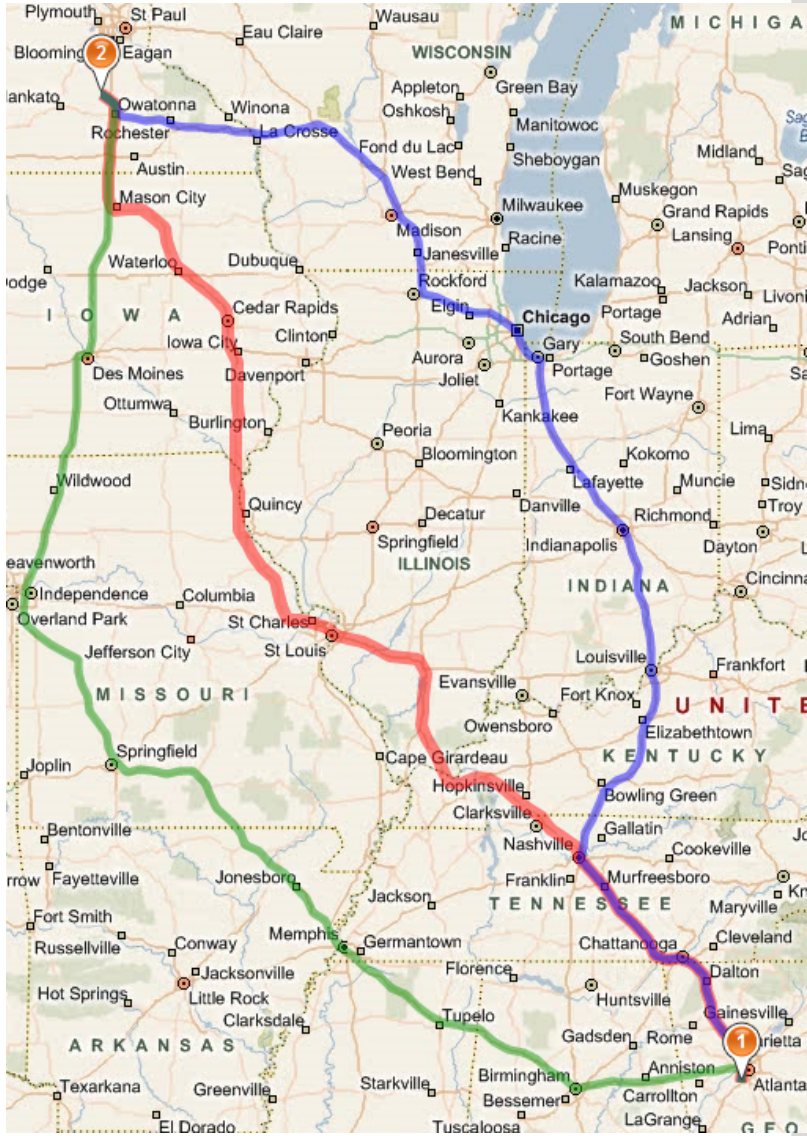
- finde gute Alternativen in einem Transportnetz

Problem:

- der kürzeste Weg ist wohl definiert
- Was aber ist eine gute Alternative?
- Problem erscheint rein heuristischer Natur
- nur auf den ersten Blick

Ziele:

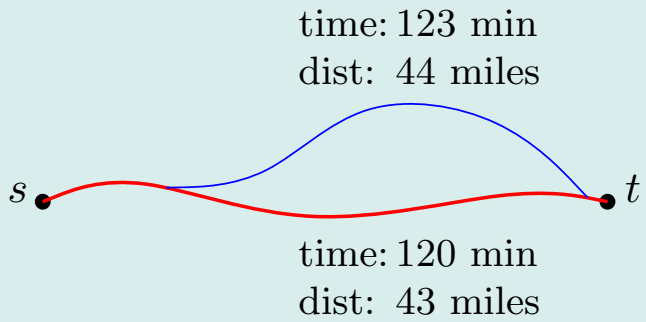
- lieber keine als schlechte Alternativen zeigen
- sollte nicht deutlich langsamer als Punkt-zu-Punkt sein



Ideensammlung

Erste Ansätze

Optimiere andere Metriken

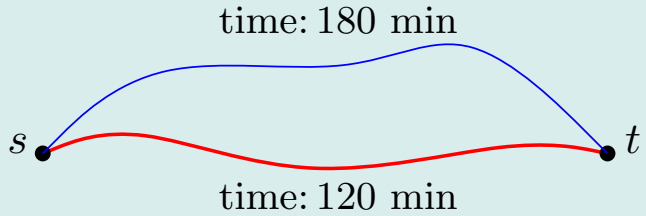


⇒ verfehlt interessante Alternativen

Berechne k-kürzeste Wege

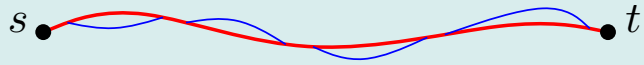
⇒ Alternative wahrscheinlich nicht unter den ersten 1000+ Pfaden

Disjunkte Pfade



⇒ verfehlt interessante Alternativen

++Gewicht auf kürzesten Weg



⇒ sieht seltsam aus

Intuition

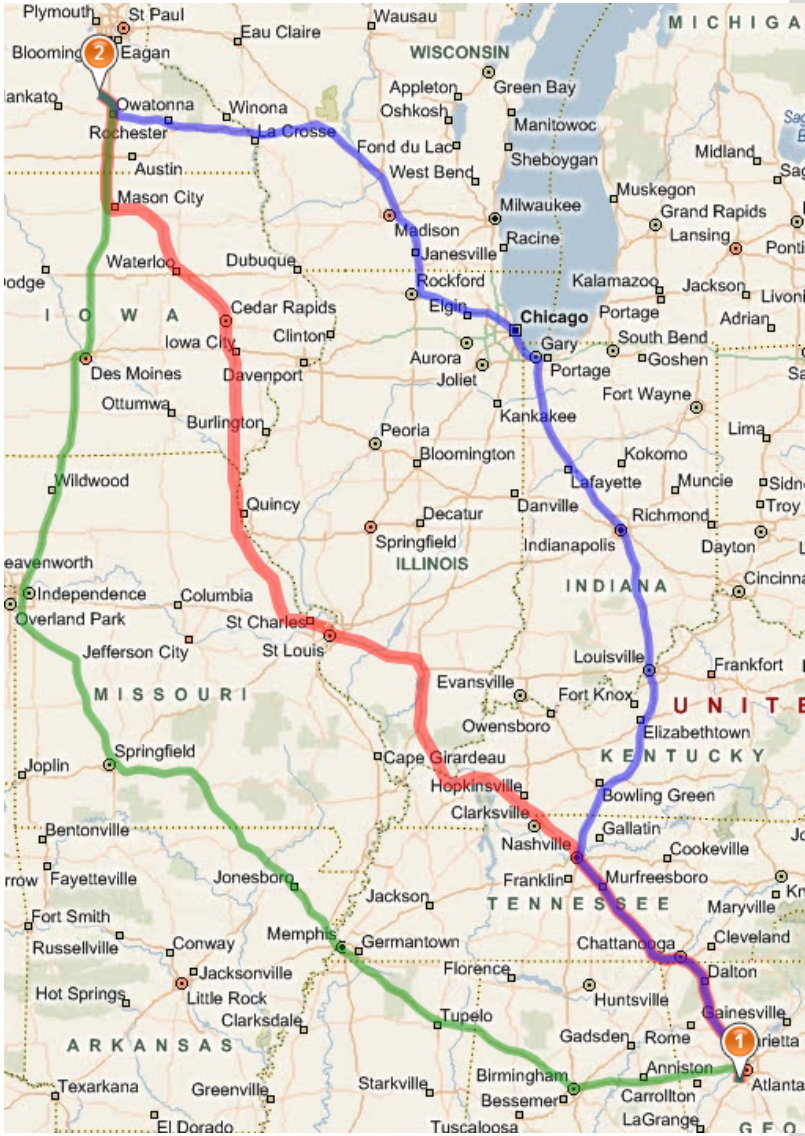
Alternativen sollten:

- nicht viel länger als der schnellste Weg sein
- signifikant verschieden

Erste Idee:

- finde einen Pfad, der Länge und Gemeinsamkeit minimiert
  - maixmal x% länger
  - teilt maximal y%

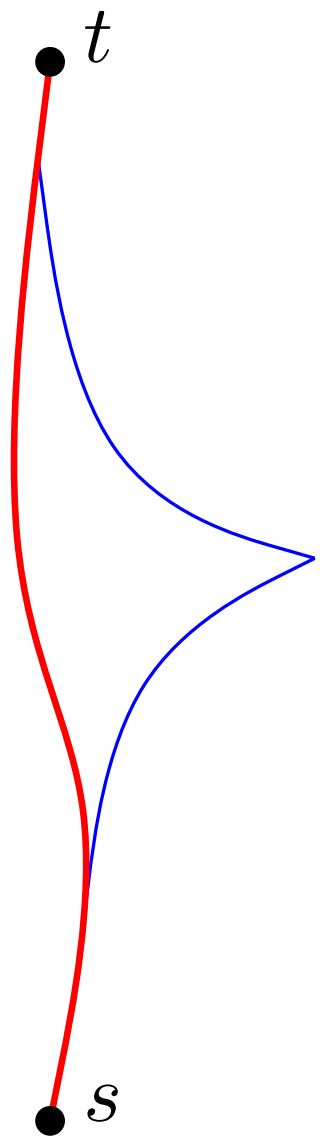
Ist das genug?



Das dritte Kriterium

Problem:

- Routen können seltsam aussehen
- sogar wenn sie verschieden und kurz sind
- lokale Umwege
- User denkt sich “das kann ich besser”



Das dritte Kriterium

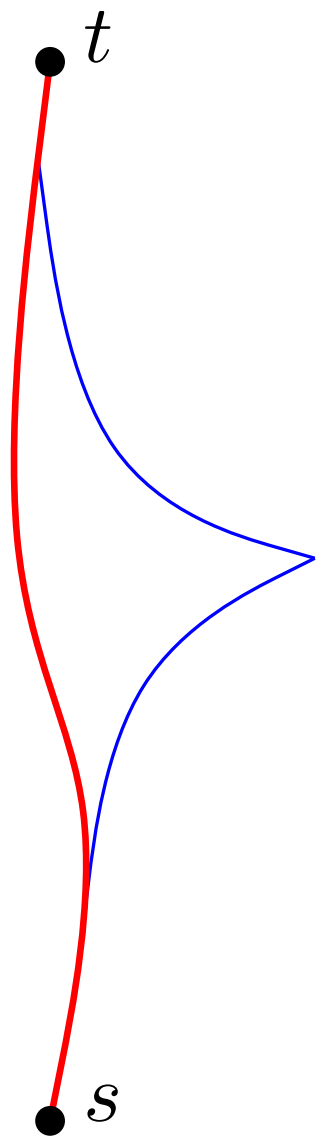
Problem:

- Routen können seltsam aussehen
- sogar wenn sie verschieden und kurz sind
- lokale Umwege
- User denkt sich “das kann ich besser”

Idee:

- kurze Subpfade müssen kürzeste Pfade sein
- beliebige Paare von Knoten auf dem Pfade sollen kleinen stretch haben

$\Rightarrow$ keine unnötigen lokalen Umwege



Das dritte Kriterium

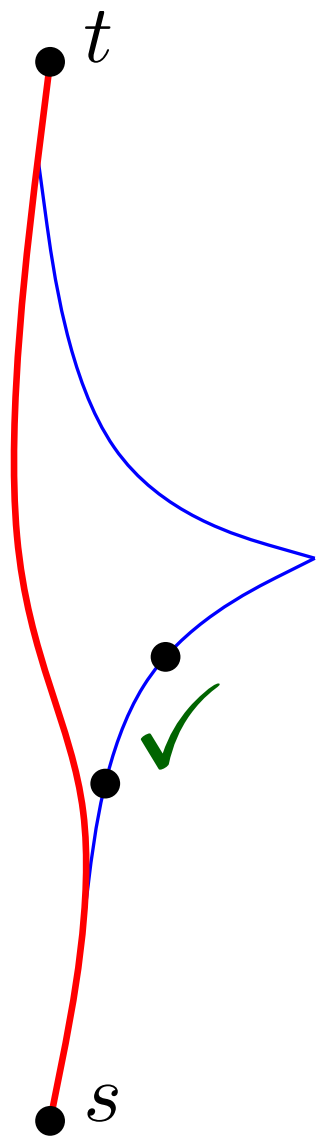
Problem:

- Routen können seltsam aussehen
- sogar wenn sie verschieden und kurz sind
- lokale Umwege
- User denkt sich “das kann ich besser”

Idee:

- kurze Subpfade müssen kürzeste Pfade sein
- beliebige Paare von Knoten auf dem Pfade sollen kleinen stretch haben

$\Rightarrow$ keine unnötigen lokalen Umwege



Das dritte Kriterium

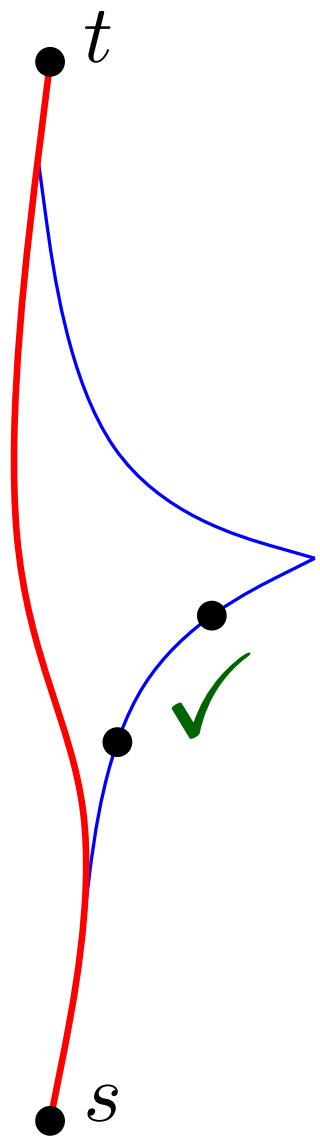
Problem:

- Routen können seltsam aussehen
- sogar wenn sie verschieden und kurz sind
- lokale Umwege
- User denkt sich “das kann ich besser”

Idee:

- kurze Subpfade müssen kürzeste Pfade sein
- beliebige Paare von Knoten auf dem Pfade sollen kleinen stretch haben

$\Rightarrow$ keine unnötigen lokalen Umwege



Das dritte Kriterium

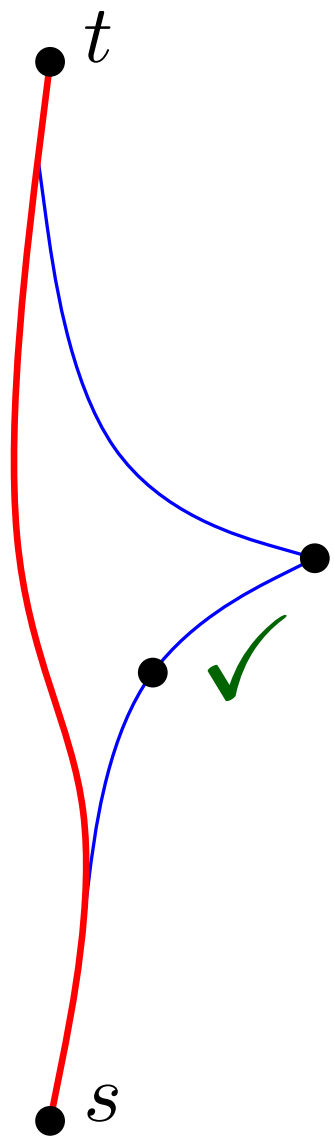
Problem:

- Routen können seltsam aussehen
- sogar wenn sie verschieden und kurz sind
- lokale Umwege
- User denkt sich “das kann ich besser”

Idee:

- kurze Subpfade müssen kürzeste Pfade sein
- beliebige Paare von Knoten auf dem Pfade sollen kleinen stretch haben

$\Rightarrow$ keine unnötigen lokalen Umwege



Das dritte Kriterium

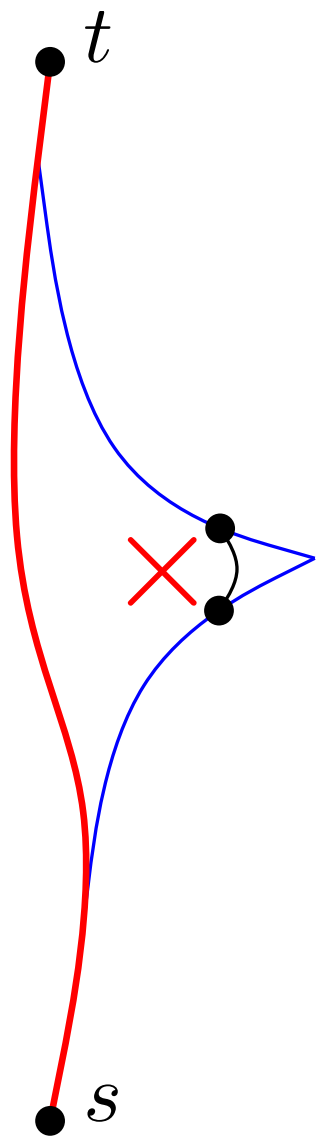
Problem:

- Routen können seltsam aussehen
- sogar wenn sie verschieden und kurz sind
- lokale Umwege
- User denkt sich “das kann ich besser”

Idee:

- kurze Subpfade müssen kürzeste Pfade sein
- beliebige Paare von Knoten auf dem Pfade sollen kleinen stretch haben

$\Rightarrow$ keine unnötigen lokalen Umwege



Eigenschaften eines Pfades P

- **Gemeinsamkeit:**
  der gemeinsame Teil von P und dem kürzestem Pfad Opt

$$\sigma(P) := \ell(Opt \cap P)$$

- **Lokale Optimalität:**
  für alle $u, v \in P$: wenn $\ell(P_{uv}) < lo(P)$, dann ist P_{uv} ein kürzester Pfad

$$lo(P) := \min_{u,v \in P} \{\ell(P_{uv}) \mid P_{uv} \text{ ist kein kürzester Pfad}\} + 1$$

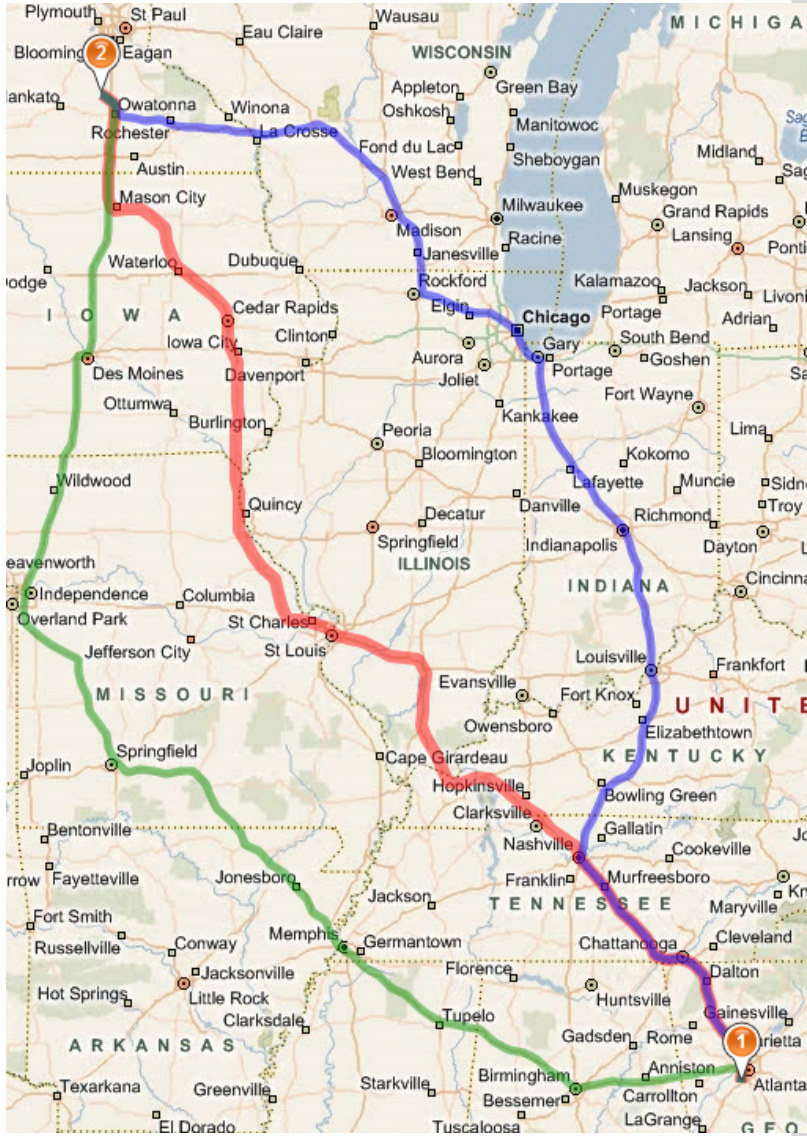
- **uniformly bounded stretch:**
  der stretch zwischen beliebigen zwei Knoten auf P muss klein sein

$$ubs(P) := \max_{u,v \in P} \{\ell(P_{uv})/\mathrm{dist}(u, v)\}$$

Gute Alternativen

ein gute Alternative *P*

- hat wenig Gemeinsamkeit mit dem kürzesten Weg *Opt*
- hat hohe lokale Optimalität
- hat niedrigen uniformly bounded stretch (UBS)



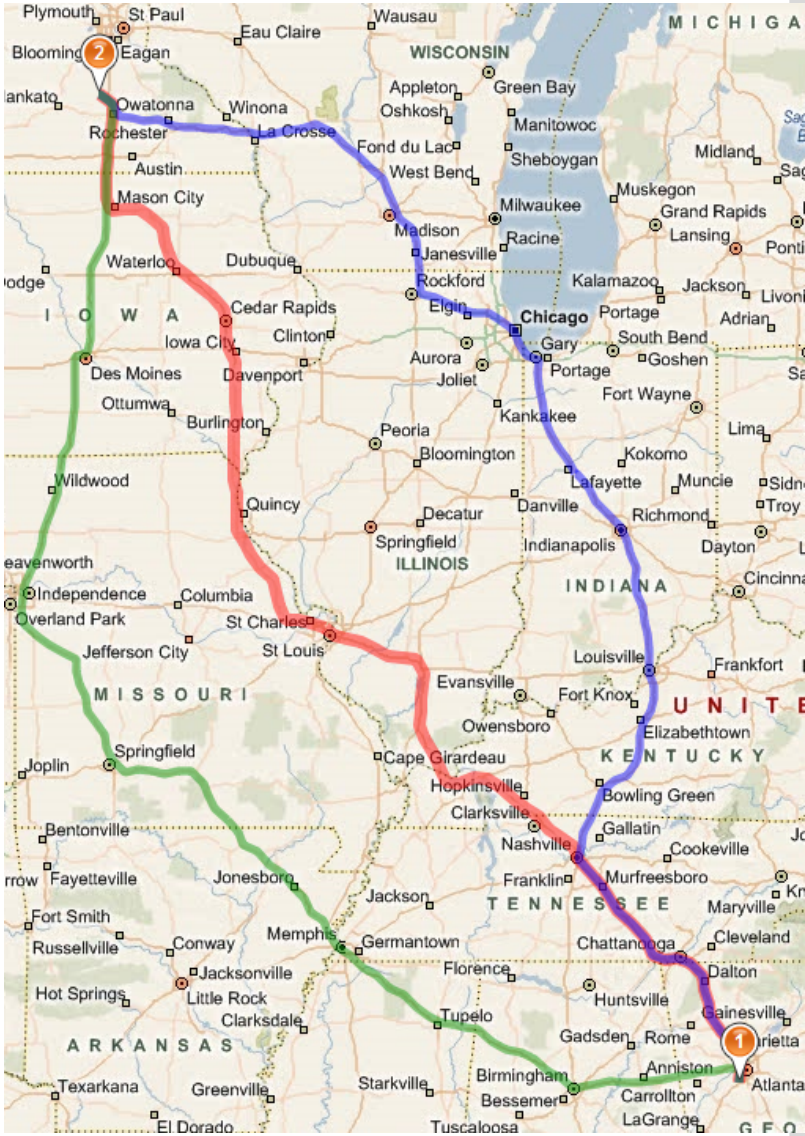
Gute Alternativen

ein gute Alternative *P*

- hat wenig Gemeinsamkeit mit dem kürzesten Weg *Opt*
- hat hohe lokale Optimalität
- hat niedrigen uniformly bounded stretch (UBS)

Idee:

- definiere Optimierungsfunktion *f*
  - lineare Kombination der drei Maße
- finde *P* mit optimalem *f* und
  - *P* hat maximal *x*% Gemeinsamkiet mit *Opt*
  - *P* hat mindestens *y*% lokale Optimalität
  - *P* hat einen UBS von maximal *z*%



Einschränkung der Möglichkeiten

problems:

- hohe Anzahl von Pfaden
- effiziente Berechnung von lokaler Optimalität and UBS?
  - $|P|$ Dijkstra Suchen
  - $|P|^2$ p2p Anfragen

  $\Rightarrow$ zu langsam

Einschränkung der Möglichkeiten

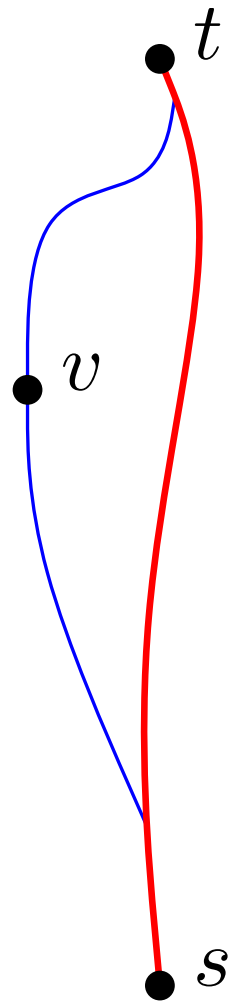
problems:

- hohe Anzahl von Pfaden
- effiziente Berechnung von lokaler Optimalität and UBS?
  - $|P|$ Dijkstra Suchen
  - $|P|^2$ p2p Anfragen

  $\Rightarrow$ zu langsam

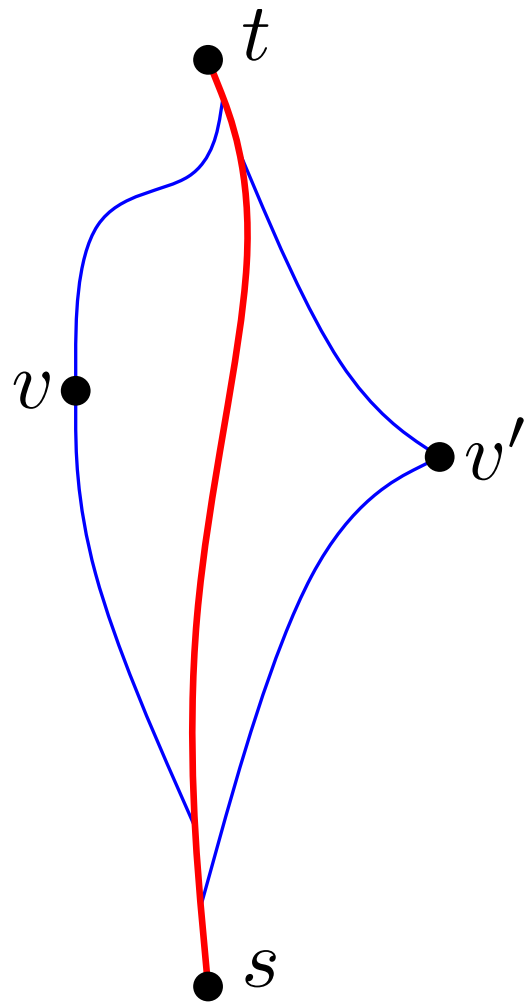
Single via paths:

- Konkatenation von zwei kürzesten Pfaden: s–v und v–t
- Eigenschaften:
  - lineare Anzahl Pfade
  - P_v ist definiert durch via Knoten v
  - lokal optimal von s nach v und v nach t
  - müssen uns nur um den Bereich um v kümmern
  - wir können für UBS was zeigen
  - können effizient berechnet werden



Single Via Paths

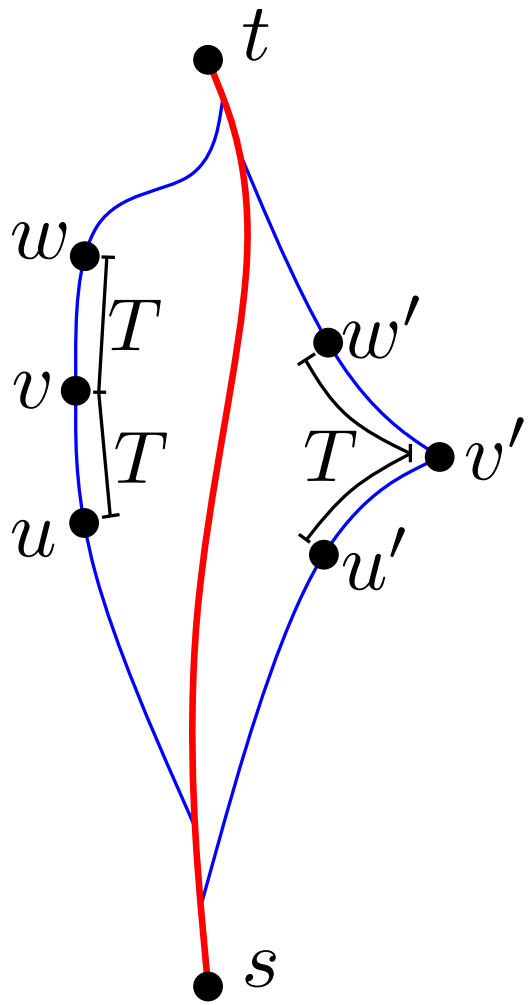
2-Approximation von lokaler Optimalität:



Single Via Paths

2-Approximation von lokaler Optimalität:

- wähle zwei Knoten $u, w \in P_v$ die T vor und nach v sind

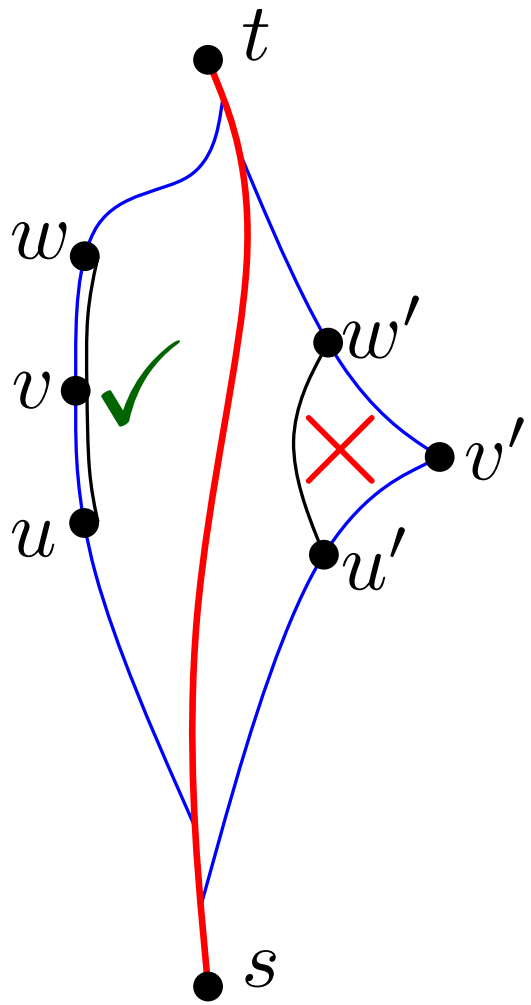


Single Via Paths

2-Approximation von lokaler Optimalität:

- wähle zwei Knoten $u, w \in P_v$ die T vor und nach v sind
- checke ob der kürzeste Pfad von u nach w v enthält
  - **JA:** P_v ist T-lokal optimal
  - **NEIN:** P_v ist nicht $2T$-lokal optimal

$\Rightarrow$ schneller Test für lokale Optimalität



Uniformly Bounded Stretch

Lemma (Uniformly Bounded Stretch)

Wenn $\ell(P_v) = (1+\epsilon)\text{dist}(s,t)$ und $lo(P_v) = \beta \cdot \text{dist}(s,t)$ gilt, dann ist der UBS von P_v maximal $\beta/(\beta-\epsilon)$.

$\Rightarrow$ es reicht die Pfadlänge zu betrachten

Erweiterter Dijkstra

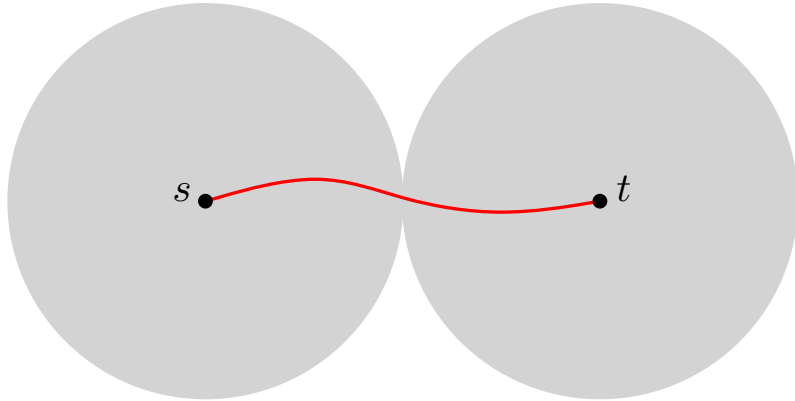
query:

- starte 2 Dijkstras
  - von s
  - nach t

Erweiterter Dijkstra

query:

- starte 2 Dijkstras
  - von s
  - nach t



Erweiterter Dijkstra

query:

- starte 2 Dijkstras
  - von s
  - nach t
- stoppe Suche wenn radii größer $(1+\epsilon)\ell(Opt)$

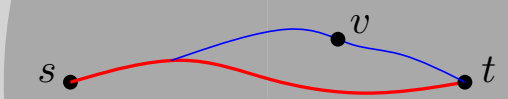
finde Alternative:

- für alle v, die von beiden Suchen gescannt worden sind:
  - berechne Pfadlänge $\ell(v)$, Gemeinsamkeit $\sigma(v)$, und mache T-test
  - entferne alle v die unsere harten constraints verletzen
- gebe P_v, der Optimierungsfunktion f minimiert, aus

Erweiterter Dijkstra

query:

- starte 2 Dijkstras
  - von s
  - nach t
- stoppe Suche wenn radii größer $(1+\epsilon)\ell(Opt)$



finde Alternative:

- für alle v, die von beiden Suchen gescannt worden sind:
  - berechne Pfadlänge $\ell(v)$, Gemeinsamkeit $\sigma(v)$, und mache T-test
  - entferne alle v die unsere harten constraints verletzen
- gebe P_v, der Optimierungsfunktion f minimiert, aus

problem:

- Anzahl der Kandidaten ist sehr hoch ($\approx 1M$)

$\Rightarrow$ Anzahl der T-tests ist $\Rightarrow$ zu langsam

RE/CH

Beobachtung:

- unwichtige Teil werden geprunt
- die wichtigen Alternativen werden wohl nicht geprunt
- reduziert Anzahl der Kandidaten

Probleme:

- Distanzen in den Suchbäumen ist inkorrekt
  - für jeden Kandidaten muss der Pfad durch 2 Queries rekonstruiert werden

RE/CH

Beobachtung:

- unwichtige Teil werden geprunt
- die wichtigen Alternativen werden wohl nicht geprunt
- reduziert Anzahl der Kandidaten

Probleme:

- Distanzen in den Suchbäumen ist inkorrekt
  - für jeden Kandidaten muss der Pfad durch 2 Queries rekonstruiert werden

RE/CH

Beobachtung:

- unwichtige Teil werden geprunt
- die wichtigen Alternativen werden wohl nicht geprunt
- reduziert Anzahl der Kandidaten

Probleme:

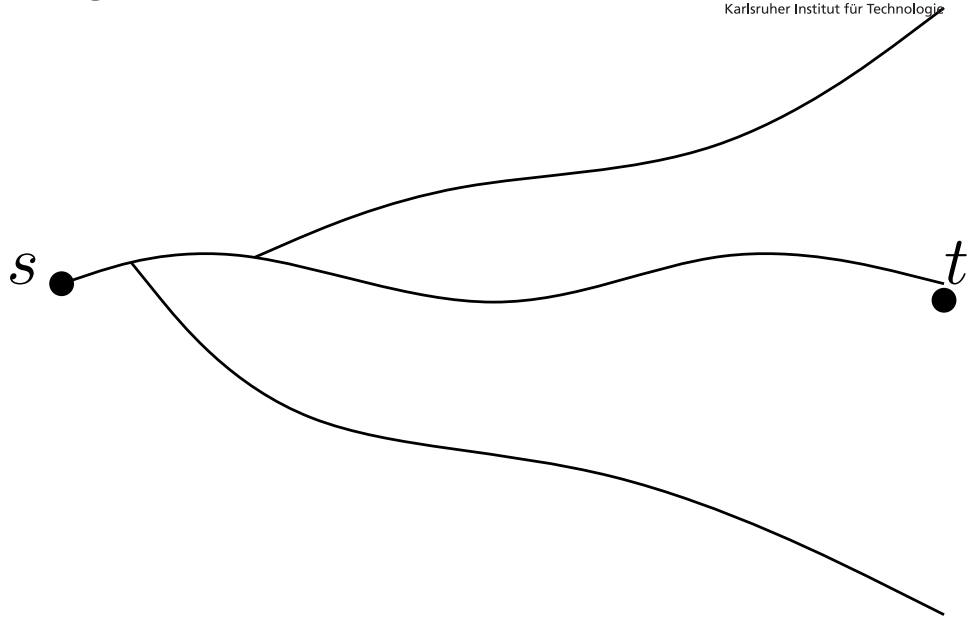
- Distanzen in den Suchbäumen ist inkorrekt
  - für jeden Kandidaten muss der Pfad durch 2 Queries rekonstruiert werden
- Anzahl Kandidaten immer noch zu hoch ($\approx 1K$)

Bounding Local Optimality

Plateaus:

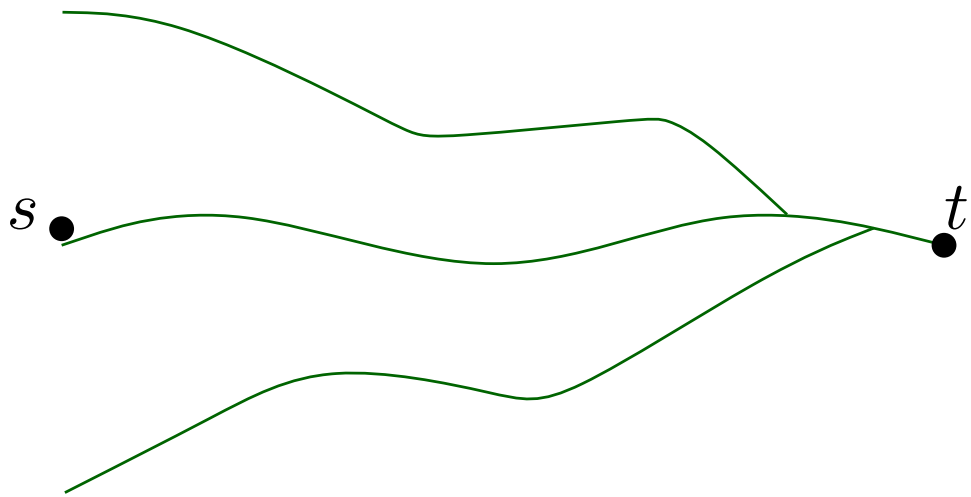
Bounding Local Optimality

Plateaus:



Bounding Local Optimality

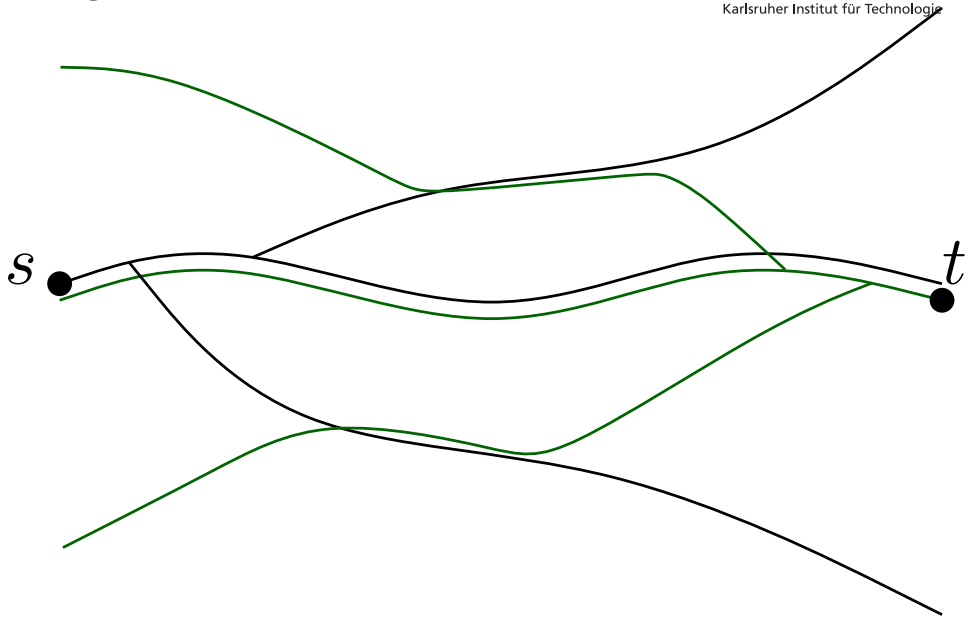
Plateaus:



Bounding Local Optimality

Plateaus:

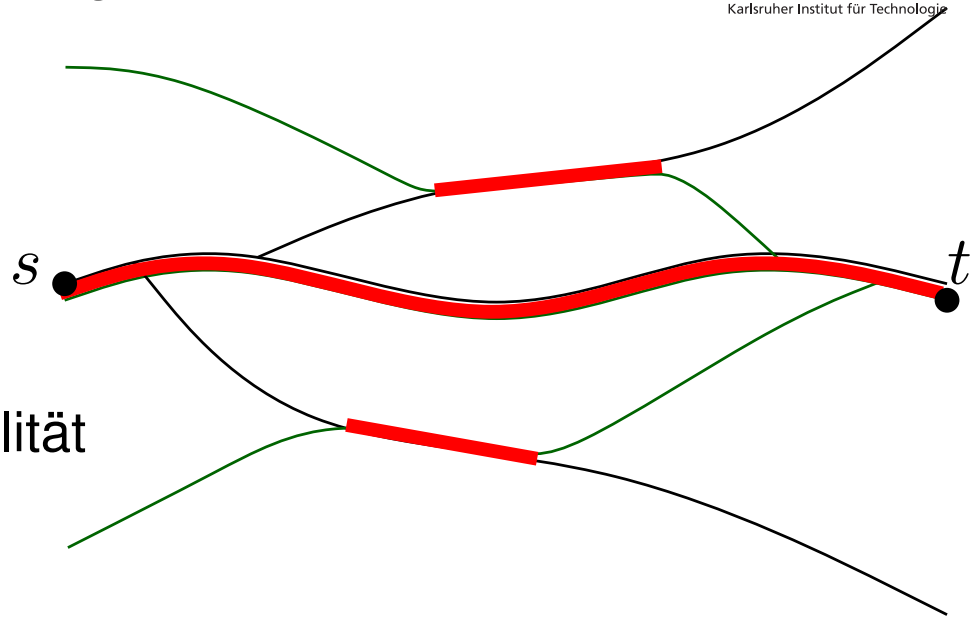
- Vorwärts- und Rückwärtsbäume schneiden sich



Bounding Local Optimality

Plateaus:

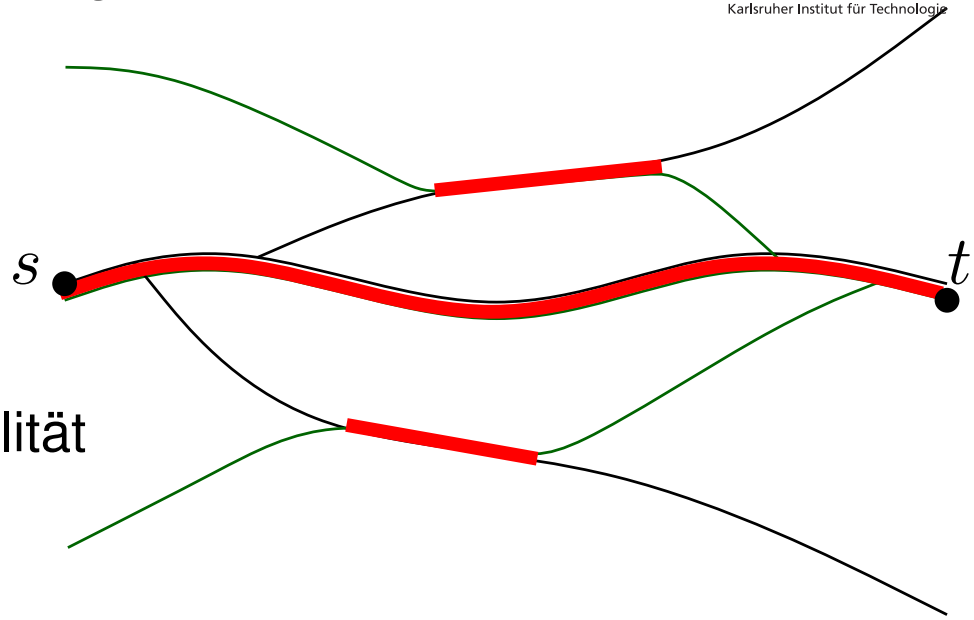
- Vorwärts- und Rückwärtsbäume schneiden sich
- definiere die Plateau-Länge $pl(v)$ für jeden Kandidaten v
- untere Schranke für lokale Optimalität
- Berechnung in linearer Zeit für alle Kandidaten möglich



Bounding Local Optimality

Plateaus:

- Vorwärts- und Rückwärtsbäume schneiden sich
- definiere die Plateau-Länge $pl(v)$ für jeden Kandidaten v
- untere Schranke für lokale Optimalität
- Berechnung in linearer Zeit für alle Kandidaten möglich



Problem

- klappt nur wenn Bäume korrekt sind
- rein heuristisch für hierarchische Techniken

Algorithmen

X–BDV:

- bidirektionale BD-Query (mit erweitertem Stoppkriterium)
- sortiere Kandidaten nach $2 \cdot \ell(v) + \sigma(v) - pl(v)$
- gebe erste Knoten aus, der Nebenbedingungen erfüllt

Algorithmen

X–BDV:

- bidirektionale BD-Query (mit erweitertem Stoppkriterium)
- sortiere Kandidaten nach $2 \cdot \ell(v) + \sigma(v) - pl(v)$
- gebe erste Knoten aus, der Nebenbedingungen erfüllt

X–REV/ X–CHV:

- bidirektionale RE/CH-Query (mit erweitertem Stoppkriterium)
- sortiere Kandidaten nach $2 \cdot \ell(v) + \sigma(v) - pl(v)$
- in dieser Sortierung:
  - rekonstruiere richtigen Pfad durch v
  - mache T-test
- gebe erste Knoten aus, der Nebenbedingungen erfüllt

$\Rightarrow$ nur 1–2 T-tests

Erhöhen der Erfolgsrate

Beobachtung:

- pruning entfernt eventuell gute Alternativen
- reach sagt aus, was der laengste kürzeste Pfad durch v ist

Erhöhen der Erfolgsrate

Beobachtung:

- pruning entfernt eventuell gute Alternativen
- reach sagt aus, was der laengste kürzeste Pfad durch v ist

Idee:

- multipliziere alle reach-werte
- vergrößert den Suchraum

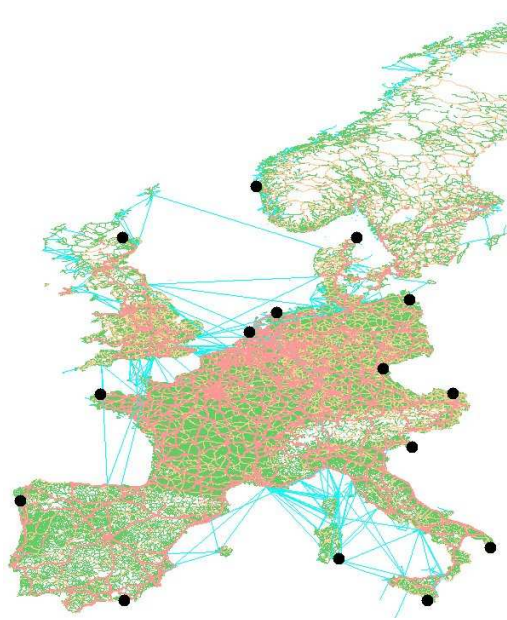
$\Rightarrow$ mehr Alternativen

$\Rightarrow$ langsamer

Experimente

input:

- Straßennetzwerk von Westeuropa
- 18 mio. Knoten
- 42 mio. Kanten



Ergebnisse:

		PATH QUALITY							PERFORMANCE		
		success	UBS[%]		sharing[%]		locality[%]		#scanned	time	slow-
p	algo	rate[%]	avg	max	avg	max	avg	min	vertices	[ms]	down
1	X-BDV	94.5	9.4	35.8	47.2	79.9	73.1	30.3	16 963 507	26 352.0	6.0
	X-REV	91.3	9.9	41.8	46.9	79.9	71.8	30.7	16 111	20.4	5.6
2	X-BDV	81.1	11.8	38.5	62.4	80.0	71.8	29.6	16 963 507	29 795.0	6.8
	X-REV	70.3	12.2	38.1	60.3	80.0	71.3	29.6	25 322	33.6	9.2
3	X-BDV	61.6	13.2	41.2	68.9	80.0	68.7	30.6	16 963 507	33 443.0	7.7
	X-REV	43.0	12.8	41.2	66.6	80.0	74.9	33.3	30 736	42.6	11.7

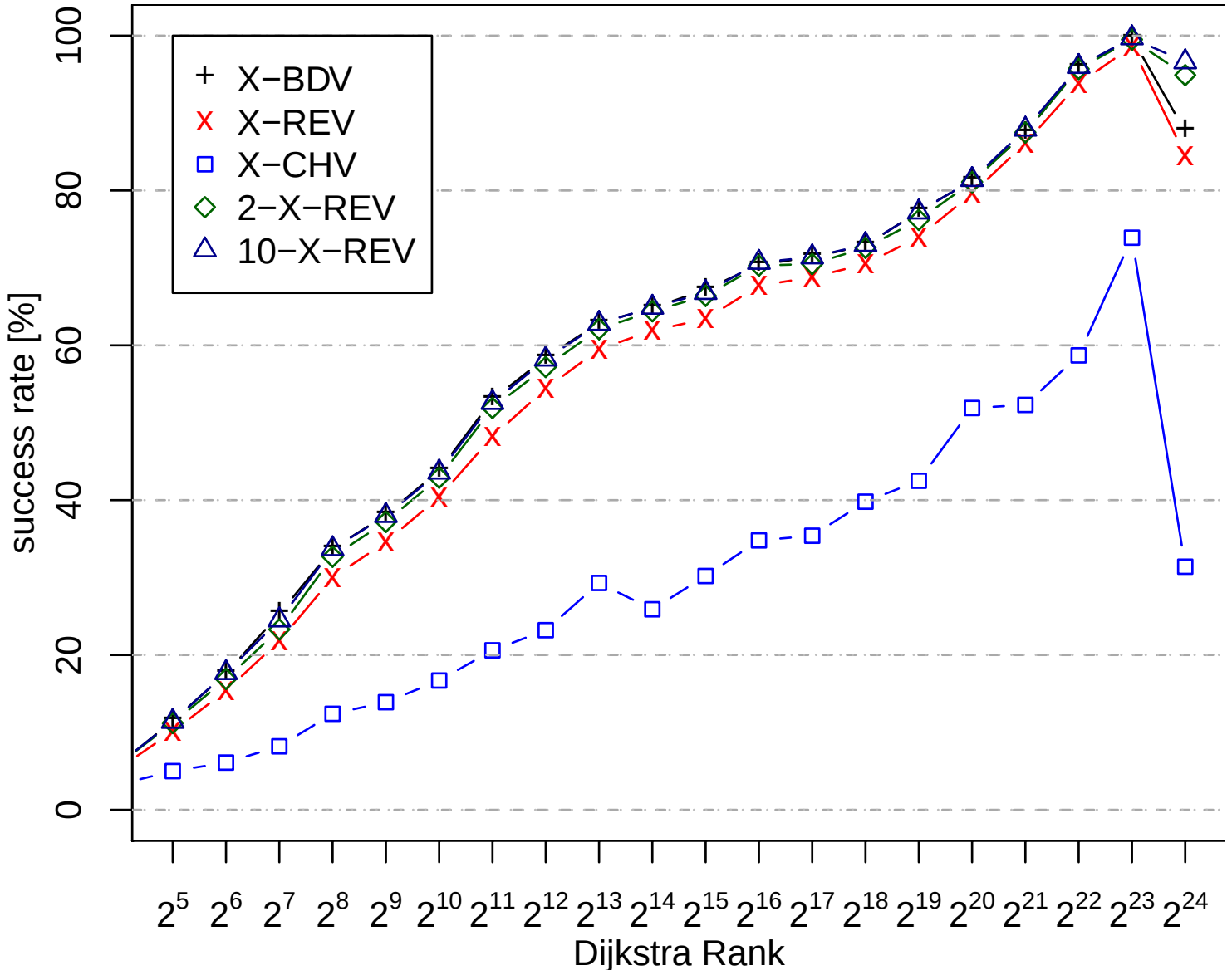
Erhöhung der Erfolgsrate

- δ gibt Reach-Multiplikator an

Ergebnisse:

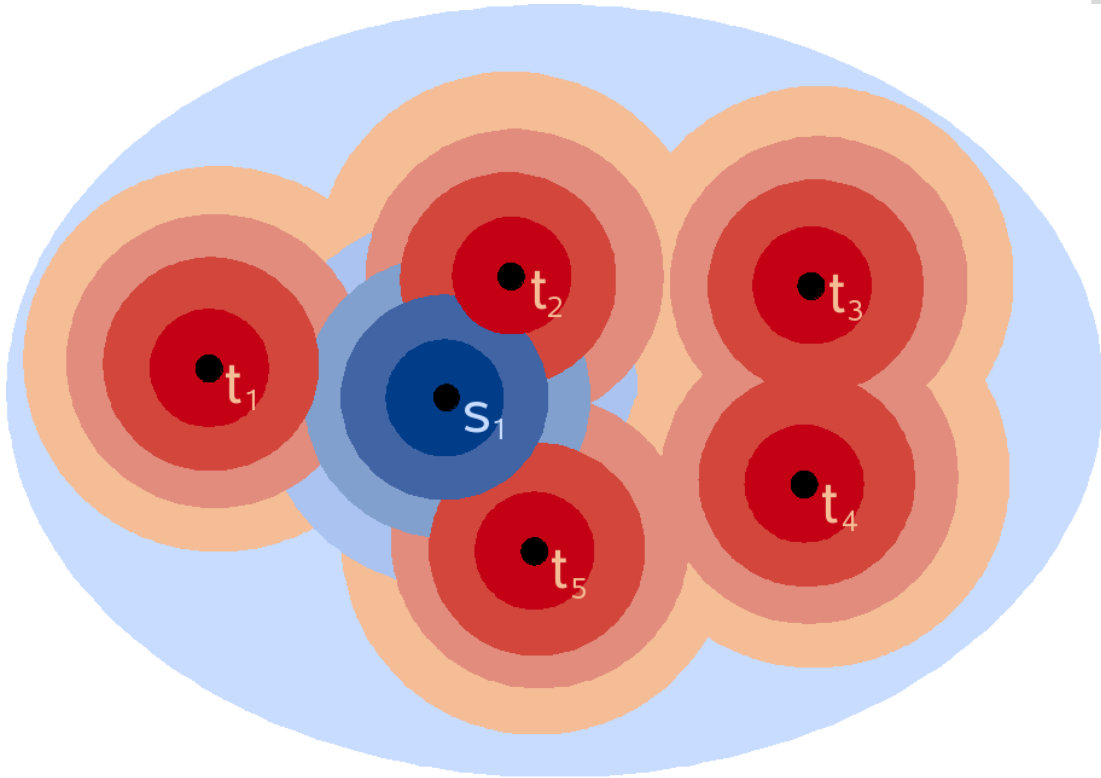
		PATH QUALITY							PERFORMANCE		
		success	UBS[%]		sharing[%]		locality[%]		#scanned	time	slow-
algo	δ	rate[%]	avg	max	avg	max	avg	min	vertices	[ms]	down
X-REV	1	91.3	9.9	41.8	46.9	79.9	71.8	30.7	16 111	20.4	5.6
	2	94.2	9.7	31.6	46.6	79.9	71.3	27.6	31 263	34.3	9.4
	3	94.2	9.5	29.2	46.7	79.9	71.9	31.2	53 464	55.3	15.2
	4	94.3	9.5	29.3	46.7	79.9	71.8	31.2	80 593	83.2	22.8
	5	94.4	9.5	29.3	46.7	79.9	71.8	31.4	111 444	116.6	31.9
	10	94.6	9.5	30.2	46.8	79.9	71.7	31.4	289 965	344.3	94.3
X-BDV	–	94.5	9.4	35.8	47.2	79.9	73.1	30.3	16 963 507	26 352.0	6.0

Lokale Queries



Zusammenfassung Many-to-Many

- berechnet Distanztabellen
- durch $|S| + |T|$ zielunabhängige hierarchische Suchen
- erzeuge Buckets $(u, d(u, t)$ während der Rückwärtssuchen
- durchsuche Buckets bei Vorwärtssuche
- pro Eintrag ca. 0.1 μs



Zusammenfassung Alternativ-Routen

- Definition von Alternativ-Pfaden
- UBS, Gemeinsamkeit, Lokale Optimalität
- effiziente Algorithmen für single via Pfade
- nur 3–5 mal langsamer als point-to-point queries

Ende

Nächste Vorlesung:
Freitag, 27. Mai, 9:45 Uhr

Literatur:

- Sebastian Knopp, Peter Sanders, Dominik Schultes, Frank Schulz, and Dorothea Wagner:
  Computing Many-to-Many Shortest Paths Using Highway Hierarchies.
- Ittai Abraham, Daniel Delling, Andrew V. Goldberg, and Renato F. Werneck:
  Alternative Routes in Road Networks.

Username: `routePlanning` Passwort: `ss10`